uc3m | Universidad Carlos III de Madrid

Grado Universitario en Ingeniería en Tecnologías Industriales
Curso 2018-2019

Trabajo Fin de Grado

"Introducción a la ciencia de datos en la Ingeniería Industrial"

Germán Guisado López

Tutor

Pablo Martínez Olmos

Leganés, Junio 2019

RESUMEN

La ciencia de datos es uno de los campos de la ciencia más en boga en la actualidad, y su importancia ha crecido ligada al desarrollo de la informática, así como al gran incremento de los datos disponibles en múltiples ámbitos. Entre sus aplicaciones más importantes se encuentra el reconocimiento de imágenes y el marketing; mientras que en la ingeniería se emplea para modelar sistemas, optimizar recursos, inteligencia artificial, etc.

El principal objetivo de este proyecto es el aprendizaje de las bases teóricas, matemáticas y estadísticas sobre los que se asienta la ciencia de datos, así como el de exponer los métodos más comunes y el tipo de problemas que son capaces de solucionar. El segundo objetivo será el de familiarizarse con algunos de los lenguajes de programación utilizados en este ámbito, en concreto, Python y MATLAB.

Finalmente, los conocimientos aprendidos serán puestos en práctica para el estudio de varias bases de datos sobre casos reales. Según el tipo de problema se aplicarán distintos algoritmos, y posteriormente se comparará la efectividad de cada algoritmo según sus resultados.

Palabras clave: Machine learning, Data analysis, Supervised learning, Information processing, Neural network

AGRADECIMIENTOS

Me gustaría dar las gracias a mi familia, ya que sin su apoyo no podría haber acudido a la Universidad, y cuya ayuda me ha permitido seguir adelante en todo momento.

Gracias a mis amigos, cuya presencia ha sido la mejor parte de la experiencia universitaria.

Gracias también a Pablo, mi tutor, por su amabilidad y ayuda para realizar este trabajo.

Y especialmente, me gustaría dedicarle este trabajo a mi abuelo Diego, con quien tantas horas pasé durante la infancia, viendo trenes y jugando al ajedrez; y que lamentablemente ya no está entre nosotros.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

1.1. Motivación del Trabajo

La motivación de este proyecto es la de complementar la formación académica como ingeniero industrial, a través del estudio de la Ciencia de Datos. Además de ser una de las disciplinas con mayor capacidad de innovación, su rápida expansión en la industria ha generado una creciente demanda de profesionales con experiencia en este campo. En concreto, un estudio realizado por la empresa *Hired* menciona que en el último año se ha incrementado la demanda de ingenieros de datos en un 38%. [2]

Una de las funciones principales de la Ingeniería es la de encontrar soluciones para diversos problemas, tanto en el ámbito académico como en el industrial. Por este motivo, la ciencia de datos es un complemento frecuentemente necesario, ya que aporta métodos capaces de resolver problemas físicos, generar modelos para optimizar procesos industriales, así como aplicaciones más llamativas como la clasificación de imágenes o el reconocimiento de voz, cruciales en la Robótica.

Finalmente, el último objetivo de este proyecto es reforzar los conocimientos en Estadística y Programación que, pese a su importancia, no se estudian con detalle dentro del plan de estudios, y que son necesarios para poder aplicar los métodos más avanzados de este proyecto.

1.2. ¿Qué es la ciencia de datos?

Tanto el término ciencia de datos como el similar *Big Data* (Datos masivos) se han convertido en palabras de moda, siendo aplicados de forma indiscriminada y a veces incorrecta. Desde el punto de vista de la ciencia, podemos definir como ciencia de datos a la extracción de conocimientos a través de la manipulación de bases de datos.

La ciencia de datos tiene múltiples aplicaciones, y generalmente no se limita a explicar y encontrar correlaciones entre los datos existentes, sino que también tiene el objetivo de generalizar los conocimientos hallados para realizar predicciones sobre el futuro.

En este trabajo, principalmente se estudian los métodos de aprendizaje automático y de predicción, mientras que otras facetas, como el manejo de bases de datos o la representación visual, se ven con menos detalle.

1.3. Marco Regulador

Con el aumento en la recopilación de datos a través de aplicaciones móviles y plataformas online, ha sido necesario debatir las consideraciones éticas y legales sobre el uso y almacenamiento de información privada. En general, esto ha llevado a la actualización y creación de nuevas leyes de protección de datos, tanto a nivel nacional (Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales. [3]), como europeo (General Data Protection Regulation [4]).

Las bases de datos utilizadas en este proyecto pertenecen a un banco de datos abierto que autoriza su utilización para fines académicos, y por tanto su uso en este trabajo cumple todos los requisitos legales. Estas son:

- Fashion-MNIST [22]
- Concrete Compressive Data Set [18]
- HTRU2 Data Set [20]
- Breast Cancer Wisconsin (Diagnosis) Data Set [41]

El software utilizado también cumple los requisitos legales, y se aprecian dos tipos: MATLAB R2016b con licencia académica para estudiantes, y software de código abierto, que incluye el lenguaje de programación Python y librerías de uso abierto como Pytorch, Matplotlib, Pandas, Scikit-learn y Numpy.

1.4. Entorno socio-económico

El presupuesto de elaboración del TFG ha sido de 75€ (invertidos en el curso de Machine Learning de Stanford, ofrecido en Coursera). El resto del material utilizado no ha supuesto ningún coste.

El impacto económico de la ciencia de datos se refleja en su crecimiento y utilidad en varias industrias, como en la de fabricación, de la que se dice que: “Data science is said to change the manufacturing industry dramatically. Let’s take under consideration several data science use cases in manufacturing that have already become common and brought benefits to the manufacturers.” [30] Entre sus usos en la industria de fabricación se incluye la predicción de fallas, optimización de precios, previsión de la demanda, el desarrollo de nuevos productos y la implementación de automatización y robotización.

La capacidad de optimización de la ciencia de datos también tiene influencia en campos como el abastecimiento de agua: “The Internet of Water – through next-generation collection devices and predictive analytics – has ushered in the ability to funnel disparate data into a single, meaningful snapshot of the entire water ecosystem.”; [31] y la ciberseguridad: “Cue cybersecurity’s newest hero: the data scientist. There is a need for data-driven solutions to cybercrime. A recent report from Indeed showed a 29% increase in demand for data scientists year over year, and a 344% increase since 2013.” [32]

Este impacto también se observa en las inversiones realizadas por empresas líderes en su sector, como Orange [33] o Altran: “Altran, global leader in engineering and ER&D services, has established dedicated Data Science and Artificial Intelligence (AI) teams in France, Italy, Spain, Portugal and Germany to respond to growing client demand for data analytics talent.”. [34]

Y al igual que Orange, la empresa IBM también menciona la importancia de la ciencia de datos como factor de disrupción e innovación en el mundo laboral: “Machine learning, big data, and data science skills are the most challenging to recruit for and potentially can create the greatest disruption to ongoing product development and go-to-market strategies if not filled.” [35]

1.5. Objetivos de aprendizaje del proyecto

En esta sección se resumen los métodos, algoritmos y otras habilidades aprendidas durante la realización del proyecto. Estos son:

- Diferencias entre aprendizaje supervisado y no supervisado.

- Redes neuronales y redes neuronales convolucionales (CNNs)
- Algoritmos de regresión: regresión lineal y kernel-ridge regression.
- Algoritmos de clasificación: regresión logística y SVM.
- Algoritmos de clustering: k-means clustering
- Algoritmos de compresión de datos (PCA) y detección de anomalías (k-NN outlier)
- Métodos para prevenir overfitting: regularización, normalización, separación de base de datos en subconjuntos (test set y training set)
- Lenguajes de programación: Python y Matlab
- Métodos para comparar calidad de modelos: Precision-Recall Curve, funciones de error, matriz de confusión.

2. MÉTODOS

2.1. Estado del Arte

En este capítulo se describirán los métodos y herramientas aplicados en los casos prácticos del proyecto. Principalmente se describirán los algoritmos utilizados, aunque añadiendo una mención a los recursos de programación empleados para el tratamiento previo de las bases de datos. Para aprender la teoría de estos métodos se han utilizado los cursos de Intro to Deep Learning with Pytorch de Facebook [39] y Machine Learning de Stanford (Coursera) [40], así como libros incluidos en la bibliografía y otra documentación.

Dentro del aprendizaje automático se distinguen dos tipos de tarea: *Supervised Learning* (aprendizaje supervisado) y *Unsupervised Learning* (aprendizaje no supervisado). Tienen distintas aplicaciones, y generalmente, los algoritmos y técnicas utilizados en cada tipo de aprendizaje son diferentes. Sin embargo, algunos métodos como las *Neural Networks* (redes neuronales) son comunes para ambas tareas, y según el objetivo deseado se ajustarán los detalles. A continuación, explicaremos en qué consiste cada tipo de aprendizaje, así como sus aplicaciones y los algoritmos más populares.

2.2. Supervised Learning

Es el caso más común en el aprendizaje automático, y se caracteriza por utilizar ejemplos que poseen una *ground truth* (es decir, que se conoce previamente la respuesta correcta). En términos matemáticos, el objetivo del aprendizaje supervisado es el de deducir una función F dependiente de un *input* o 'x' (datos de entrada) para predecir un *output* (resultado) 'y'. Es decir:

$$y = F(x) \tag{2.1}$$

Como el resultado para cada dato de entrada es conocido, el algoritmo usa un método iterativo, comparando los resultados verdaderos con las predicciones, y ajustando la función F para lograr una mayor precisión.

Dentro del aprendizaje supervisado se diferencian dos tipos de problema, los de regresión y los de clasificación.

2.2.1. Problemas de Regresión

El análisis de regresión es uno de los procesos estadísticos más comunes en el modelado estadístico, y generalmente su objetivo es hallar una función continua (es decir, en el que la variable dependiente 'y' sea continua) a partir de una o varias variables independientes 'x' (que son los atributos de la base de datos).

Los modelos de regresión tienen una función predictiva, y también permiten analizar separadamente si existe una relación entre la variable dependiente y cada una de las variables independientes del problema. Por tanto, podemos definir la regresión de forma general como:

$$y = F(X, \beta) \tag{2.2}$$

Donde ‘y’ es la variable dependiente, ‘X’ la matriz de variables independientes, y β los parámetros desconocidos que relacionan ‘y’ con cada variable independiente. El objetivo de la regresión es por tanto hallar estos parámetros β, que determinan la influencia de cada variable.

En la actualidad, existen varias formas de solucionar los problemas de regresión, y en este proyecto se aplicarán tres:

- **Regresión Lineal**

En concreto, el método utilizado en este proyecto es una regresión lineal múltiple, y se diferencia de la regresión lineal simple en considerar varias variables independientes en vez de una. Ambos son casos particulares de la ecuación general lineal, que considera múltiples variables dependientes en vez de una. Por tanto, la ecuación de la regresión lineal múltiple se define como:

$$y_i = \beta_0 + \beta_1 x_i^{(1)} + \beta_2 x_i^{(2)} + \cdots + \beta_n x_i^{(n)} \quad (2.3)$$

Donde ‘i’ se refiere a cada una de las observaciones en la base de datos, y ‘n’ a las variables independientes o atributos.

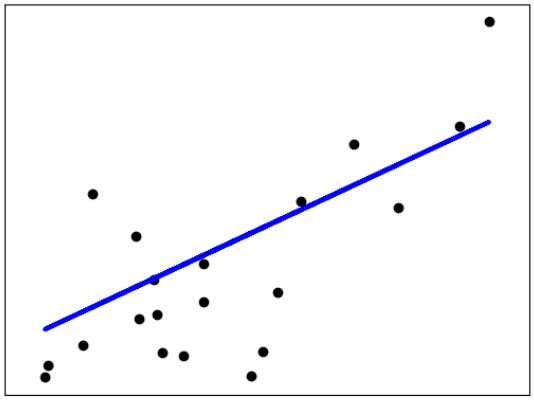

Fig. 2.1 Ejemplo de regresión lineal [5]

La calidad del modelo de regresión se analizará a través de una función de error, como por ejemplo el error cuadrático medio (ECM, llamado MSE o *Mean Square Error* en inglés), que compara la predicción del modelo (ŷ) con el valor real (y):

$$E(\beta) = \frac{1}{m} \sum_{i=1}^{m} (\hat{y} - y)^2 \quad (2.4)$$

Un modelo será más exacto cuanto menor sea el valor de la función de error, por tanto, el objetivo es encontrar los valores de β para los cuales se minimiza el error. El método más común para optimizar la función es el algoritmo *Gradient Descent* (descenso por gradiente), que actualiza de forma iterativa el valor de los parámetros β a partir del gradiente de la función de error:

$$\beta_j = \beta_j + \alpha \frac{\partial}{\partial \beta_j} E(\beta) \quad (2.5)$$

Donde ‘j’ se refiere a cada uno de los parámetros, y α es el *learning rate* (tasa de aprendizaje o tamaño de paso) que regula la sensibilidad del algoritmo. Un valor alto de α (mayor sensibilidad) acelera el proceso, pero puede causar que no se alcance el mínimo de la función a optimizar, mientras que un valor demasiado bajo hace que se requieran muchas iteraciones para alcanzar el mínimo, aumentando los costes de tiempo y computación.

- **Kernel Ridge Regression**

Es un algoritmo más avanzado que combina *Ridge Regression* (regresión de arista) y el método *Kernel* (núcleo).

Ridge Regression es una variación del método de regresión lineal diseñado para resolver el mayor defecto al que se enfrentan estos modelos, el *overfitting* (sobreajuste). El overfitting ocurre cuando el modelo ajusta sus parámetros en exceso a los datos de entrenamiento, perdiendo capacidad de generalización y poder predictivo (con respecto a los datos no utilizados). Es decir, si se separa la base de datos en dos subgrupos: datos de entrenamiento y datos de prueba, el overfitting ocurre cuando la función de error para los datos de prueba comienza a aumentar. Para prevenir el overfitting, Ridge Regression utiliza el método de regularización, que reduce el tamaño de los parámetros β mediante una penalización, lo que simplifica la hipótesis del modelo y evitando que se ajuste demasiado a los datos de entrenamiento. [6] La regularización se ajusta mediante el coeficiente λ, cuanto menor sea, menor será el efecto de la regularización (un modelo sin regularización tendrá por tanto λ=0). Al restar este coeficiente, también cambia la función de error, por lo que ahora la función que debe ser minimizada es (donde ϕ son los datos de entrada regularizados, N el número de observaciones y D es el número de variables de entrada):

$$L(\theta, \lambda) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \theta^T \phi(x_i))^2 + \lambda \sum_{j=1}^{D+1} \theta_j^2 \qquad (2.6)$$

La imagen siguiente ilustra el efecto de λ, cuanto mayor es, menores serán los parámetros β:

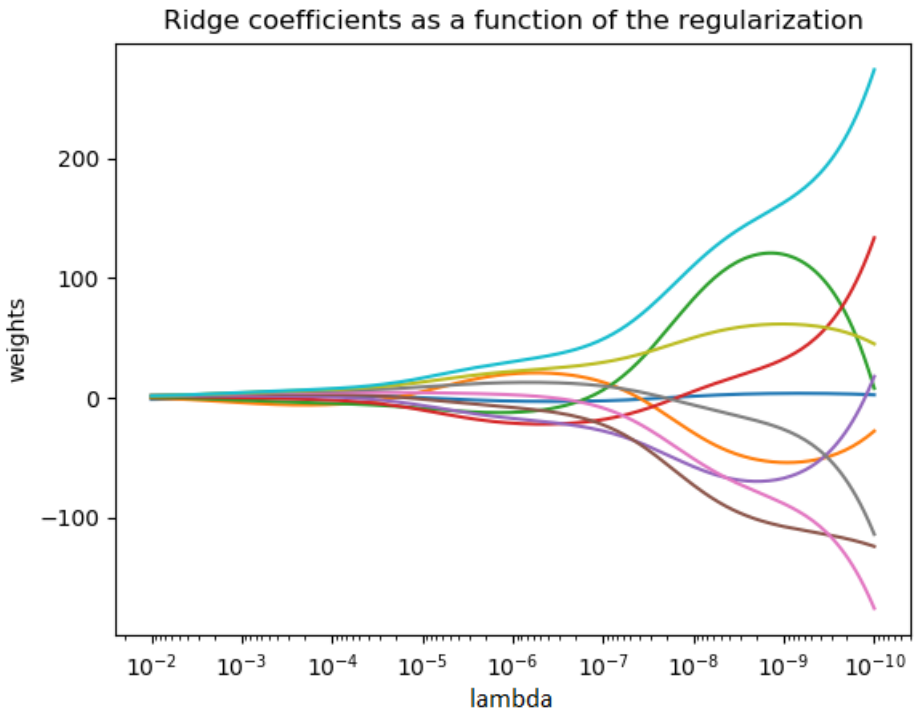

Fig. 2.2 Efecto de la regularización en los parámetros del modelo [7]

El método Kernel (también conocido como *kernel trick)* se basa en la idea de que datos que no se relacionan linealmente en una dimensión determinada pueden ser transformados a otra dimensión en la que sea más sencillo relacionarlos. Como esta transformación es complicada y tiene un coste elevado, el método kernel simplifica el proceso mediante el uso de productos internos en vez de requerir la transformación exacta de los datos. Matemáticamente, al calcular el gradiente de la función 2.6 e igualarla a cero, se obtiene:

$$\theta_\lambda = \sum_{i=1}^{N} a^i \, \phi(x^i) \quad donde \; a^i = -\frac{1}{\lambda}(y_i - \theta_i^T \phi(x_i)) \tag{2.7}$$

A partir de 'a', la predicción del modelo para un nuevo vector x* se calcula de la siguiente forma:

$$y^* = \sum_{i=1}^{N} a_\lambda^i \, k(x^i, x^*) \tag{2.8}$$

Donde 'k' es la función kernel, que debe corresponder a un producto escalar en el subespacio de la matriz de variables de entrada. Existen múltiples funciones kernel utilizadas, algunos de los ejemplos más comunes pueden verse en la referencia [36]. Una vez elegida la función k, el objetivo es encontrar el vector 'a' que minimiza la función de error de la ridge regression. Sustituyendo $\theta=\Phi^T$ se puede expresar la función de error a optimizar (siendo K la matriz kernel: $K=k(x^m,x^n)$ como:

$$L(a,\lambda) = \frac{1}{N}[a^T K^T K a - a^T K y + \lambda a^T K a] \tag{2.9}$$

En resumen, la regresión Kernel-Ridge es capaz de crear un modelo lineal a partir de los datos transformados, combinando mayor velocidad (gracias al método Kernel) y efectividad (al emplear regularización) que otros modelos más simples.

- **Redes Neuronales**

Las redes neuronales se aplican para todos los casos estudiados en este proyecto, por lo que se explicarán en detalle en el apartado 2.4.

2.2.2. Problemas de Clasificación

El objetivo de los algoritmos de clasificación es el de determinar a qué categoría pertenece cada observación. Estas categorías han de ser definidas previamente, formar parte de los datos existentes (ya que la clasificación es un método de aprendizaje supervisado) y ser mutuamente excluyentes. Según el número de categorías, los problemas de clasificación pueden ser binarios (vivo/muerto) o múltiples (gato, perro, caballo, elefante).

Por tanto, la diferencia con los algoritmos de regresión es que, en el caso de la regresión, el output buscado es discreto en vez de continuo. Sin embargo, esto no significa que la función que relaciona las variables de entrada y salida deba ser discreta. Muchas de las funciones utilizadas en el ámbito de clasificación dan la probabilidad de que una observación pertenezca a cada una de las categorías. Por ejemplo, un modelo para supervisión de e-mail indicaría que un correo concreto tiene una probabilidad de 70% de ser spam, y 30% de no serlo. Para poder realizar predicciones concretas se debe elegir una *decision boundary* (umbral de decisión). Como ejemplo, si se elige 50% como decisión boundary, el modelo predecirá que las observaciones con output superior a 50% son spam, y cuando el output sea menor a 50% predecirá que no son spam.

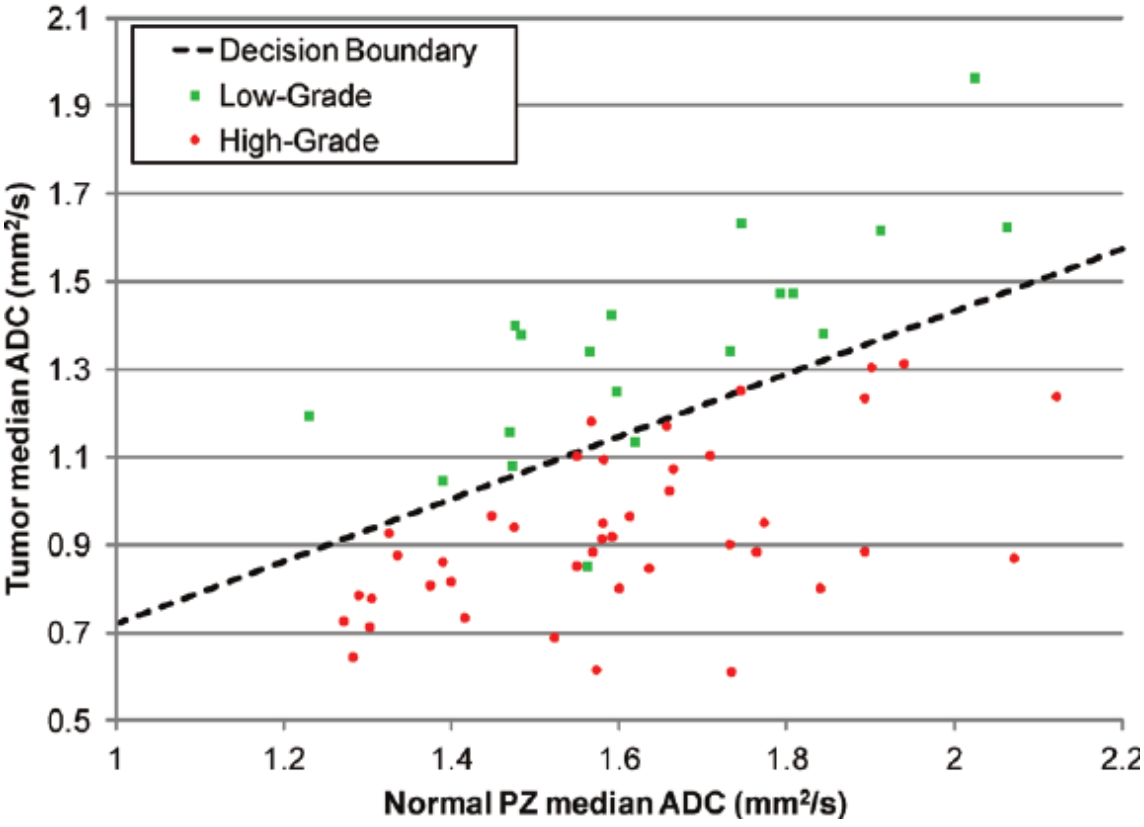

Fig. 2.3 Ejemplo de decisión boundary [8]

Entre los algoritmos más utilizados para clasificación se encuentran: *Boosting*, *Random forests, Support Vector Machines (SVM), Logistic Regression*, etc. En este proyecto se estudian los dos últimos, además de las redes neuronales.

- **Logistic Regression (Regresión Logística)**

Para los problemas de clasificación, especialmente para casos con múltiples categorías, la regresión lineal no sirve, ya que su output no nos permite medir la probabilidad de cada suceso. Para solucionarlo, se emplea la función logística, que devuelve valores entre 0 y 1, que equivalen a la probabilidad de cada suceso. Además, esta función puede ser aplicada tanto a problemas binarios como múltiples. La función logística es:

$$y(x) = \frac{1}{1+e^{-x}} \tag{2.10}$$

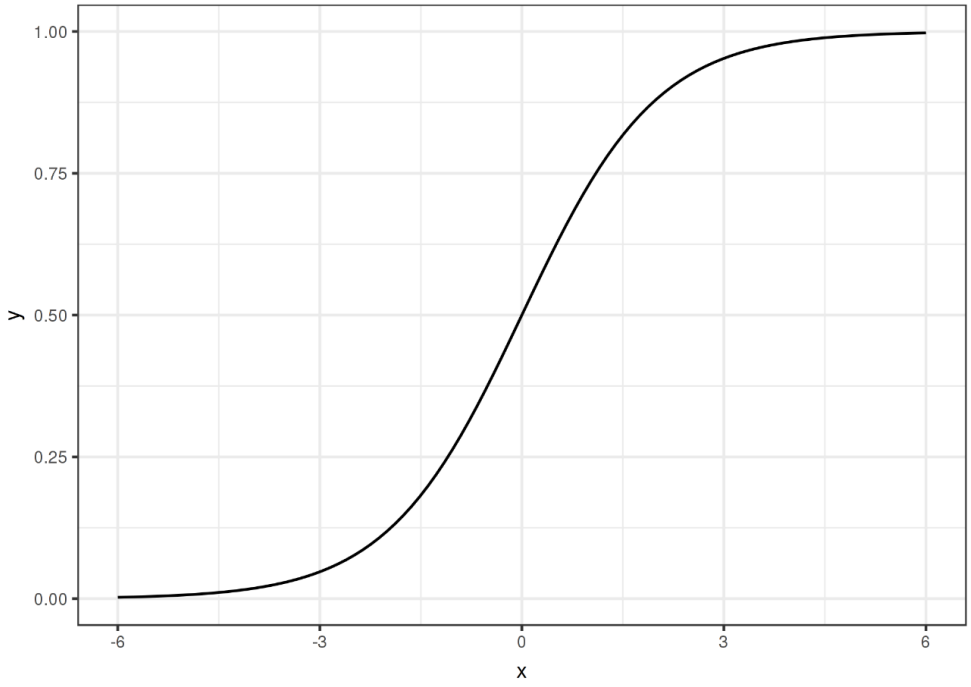

Fig. 2.4 Ejemplo de función logística [9]

Los modelos de regresión logística aplican esta función asignando parámetros β a las variables de entrada:

$$y_i = \frac{1}{1+e^{-(\beta_0+\beta_1 x_i^{(1)}+\beta_2 x_i^{(2)}+\cdots+\beta_n x_i^{(n)})}} \tag{2.11}$$

Al igual que en la regresión lineal, se utiliza una función de error para analizar la exactitud del modelo. Sin embargo, la función de error utilizada en regresión lineal no es convexa para modelos de regresión logística, y es necesario utilizar una función que si lo sea para permitir la optimización del modelo. La más común es *logistic loss*, siendo ŷ las predicciones del modelo se formula como:

$$para\ y = 1: \qquad E(\beta) = \frac{1}{m}\sum_{i=1}^{m} -\log(\hat{y})$$

$$para\ y = 0: \qquad E(\beta) = \frac{1}{m}\sum_{i=1}^{m} -\log(1-\hat{y}) \tag{2.12}$$

Para optimizar el modelo se utiliza Gradient Descent (derivando la función de coste como se mostró en el apartado de regresión lineal) u otro algoritmo de optimización más complejo, como L-BFGS o *Conjugate Gradient.*

- **Support Vector Machines (SVM)**

También conocidas como Máquinas de vectores de soporte en español, es un algoritmo potente que puede ser aplicado a la mayoría de los problemas de aprendizaje automático, pero que está principalmente adaptado para problemas de clasificación.

El objetivo del algoritmo SVM es, partiendo de un espacio de dimensión 'N' (siendo esta N el número de variables independientes del problema), encontrar un hiperplano capaz de separar (clasificar) las observaciones en dos grupos definidos. Un hiperplano es un subespacio con una dimensión menor al espacio en el que se encuentra, por ejemplo, en un espacio bidimensional (N=2), un hiperplano sería una recta; y en un espacio tridimensional (N=3), el hiperplano sería un plano de 2 dimensiones. [11]

Para los problemas de clasificación binaria, sólo se necesita un hiperplano para separar los datos. Lógicamente, para problemas de clasificación múltiple es necesario encontrar múltiples hiperplanos para separar las observaciones en el número de grupos buscados.

El hiperplano funciona como una decisión boundary, y es optimizado de forma que el margen entre el hiperplano y la observación más cercana de cada categoría es la máxima posible, como se observa en la siguiente figura:

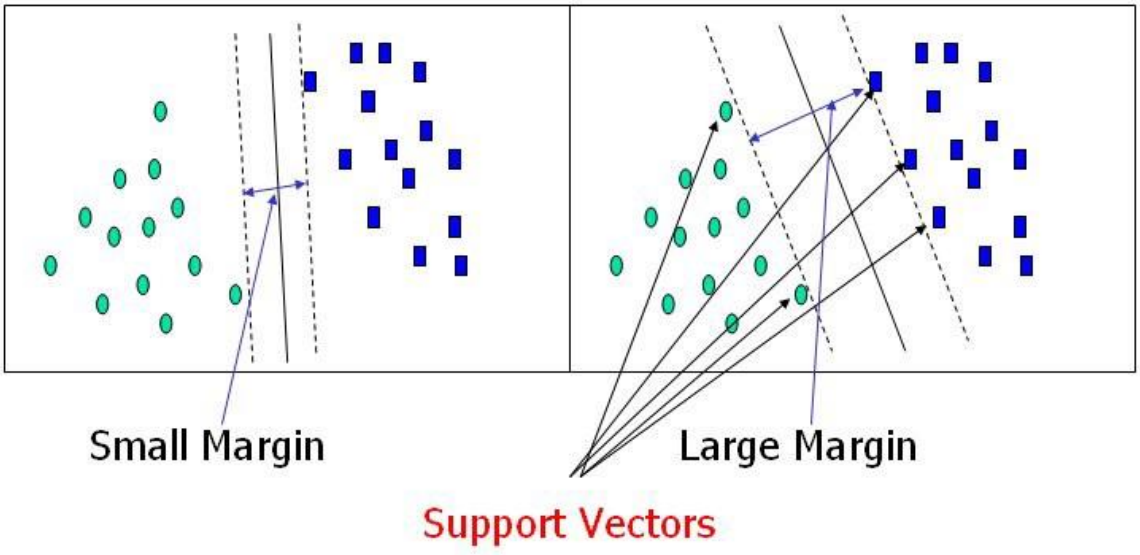

Fig. 2.5 Ejemplo márgenes en SVM [11]

Se escoge por tanto el ejemplo de la derecha, ya que el margen es mayor. Las observaciones más cercanas al hiperplano son las denominadas *support vectors* (vectores de apoyo), y tienen mayor influencia en la posición del hiperplano que el resto de las observaciones.

El desarrollo matemático de SVM es más complejo, y queda fuera de los límites de este proyecto. Se puede consultar en la referencia [37].

2.3. Unsupervised Learning

La principal diferencia entre el aprendizaje supervisado y el no supervisado es que en este último no se conoce (o no existe) una respuesta correcta. Es decir, que sólo se poseen los datos de entrada (x) pero no los de salida. Por tanto, el objetivo es el de encontrar estructuras y relaciones entre los datos, principalmente mediante la aplicación de funciones de distribución de probabilidad.

Al igual que en Supervised Learning, los problemas de aprendizaje no supervisado se agrupan mayoritariamente en tres tipos: *clustering*, compresión de datos y detección de anomalías.

2.3.1. Clustering

En español se traduce como ‘agrupamiento’ o bien ‘análisis de grupos’. Es la función más común en el aprendizaje no supervisado, y su objetivo es organizar objetos (es decir, cada ‘x’) que comparten ciertas características en grupos. Por tanto, los objetos serán similares a los demás en el mismo grupo. Estas agrupaciones pueden realizarse a través de distintas formas, por ejemplo, se puede agrupar en base a una sola característica (color, forma, tamaño) o en base a varias variables.

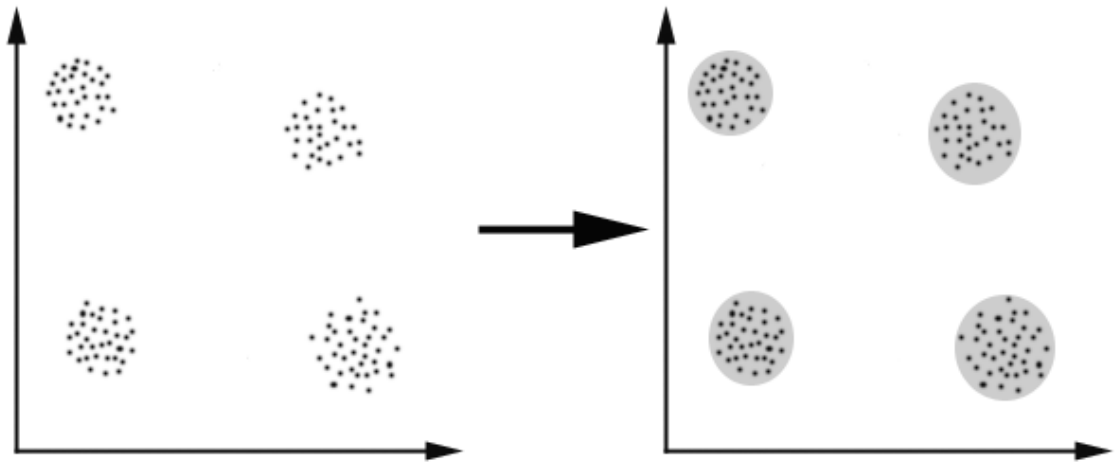

Fig. 2.6 Ejemplo de Clustering [12]

El algoritmo más utilizado es el denominado ‘*K-means Clustering’*, que requiere elegir previamente el número de grupos buscados. Cada grupo tendrá un centroide, que comienzan situados en un lugar aleatorio, y que el algoritmo irá desplazando de forma iterativa hasta separar los datos en el número de grupos deseado. Otros algoritmos populares son el DBSCAN, las redes neuronales y el agrupamiento jerárquico.

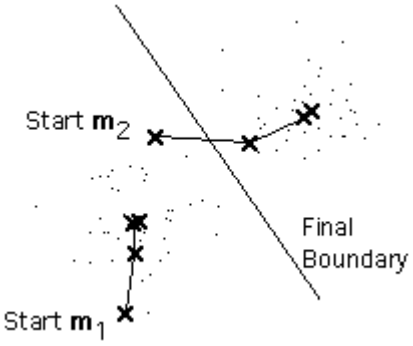

Fig. 2.7 Ilustración del método K-means [13]

El clustering tiene una gran variedad de aplicaciones, entre las que se encuentran:

- **Genética:** secuenciación de ADN, estudio de grupos de población y comparación de datos genealógicos.
- **Marketing:** estudios de mercado (búsqueda de segmentos de mercado en la población), sistemas de recomendación de productos.
- **Páginas web:** uso en redes sociales para identificar comunidades o grupos de personas afines.
- **Procesamiento de imágenes:** reconocimiento de objetos y filtrado de imágenes.
- **Organización documentos:** permite identificar spam en e-mail y separar correos según su contenido.

2.3.2. Compresión de Datos

Su objetivo es el de reducir el tamaño de los datos usados, con el fin de reducir el espacio de almacenamiento ocupado, o bien para acelerar el uso o procesamiento de esos datos mediante otros algoritmos. Generalmente, esta reducción de tamaño consiste en reducir el número de columnas (variables) de la base de datos, a base de combinar o eliminar (cuando no aportan información) variables.

El algoritmo más popular es el *Principal Component Analysis* (Análisis de componentes principales), que utiliza transformaciones ortogonales para generar nuevas variables (llamadas componentes principales) en forma de base ortogonal. Otros algoritmos comunes son el SVD (Descomposición en valores singulares) y el NNS (búsqueda de vecinos más próximos).

2.3.3. Detección de Anomalías

Consiste en identificar objetos muy diferentes a los demás de la base de datos. Aunque generalmente son problemas de aprendizaje no supervisado, también se pueden emplear técnicas de aprendizaje supervisado para este tipo de problemas.

El algoritmo más utilizado es *k-NN outlier*, una variación del método de vecinos más próximos, que generalmente se usa en problemas de regresión y clasificación. Para la detección de anomalías, se utiliza este método para obtener la densidad local de cada objeto (cuanto más lejos del vecino más próximo, menor es la densidad local), y las anomalías corresponderán con los menores valores obtenidos de densidad local.

La principal aplicación es detección de fraude, aunque también se utiliza para sensores, estudio de ecosistemas y sistemas de monitorización.

2.4. Neural Networks

Las redes neuronales artificiales son un algoritmo inspirado por la biología, que intentan imitar el funcionamiento del cerebro humano para resolver problemas. Su elemento básico son los nodos (también conocidos como neuronas artificiales), que a su vez forman capas (*layers)* interconectadas. Toda red neuronal tiene una capa de entrada, una capa de salida, y un número variable de capas ocultas (intermedias) de las que debe haber como mínimo una.

Cada nodo contiene una función de activación, con la que procesa varios inputs (procedentes de la capa anterior), produciendo un output que será transmitido a la siguiente capa. Existen muchas funciones de activación distintas, que permiten ajustar la red neuronal para diversos objetivos, y es por este motivo que las redes neuronales pueden ser utilizadas en todos los ámbitos del aprendizaje automático. Las funciones de activación más comunes son la función sigmoide o logística (para problemas de clasificación y probabilidad); la función lineal (para regresión); y la función ReLU (rectificador), que es una función lineal para valores positivos, pero cambia los valores negativos por 0, y que mejora la velocidad de computación para redes neuronales con muchas capas.

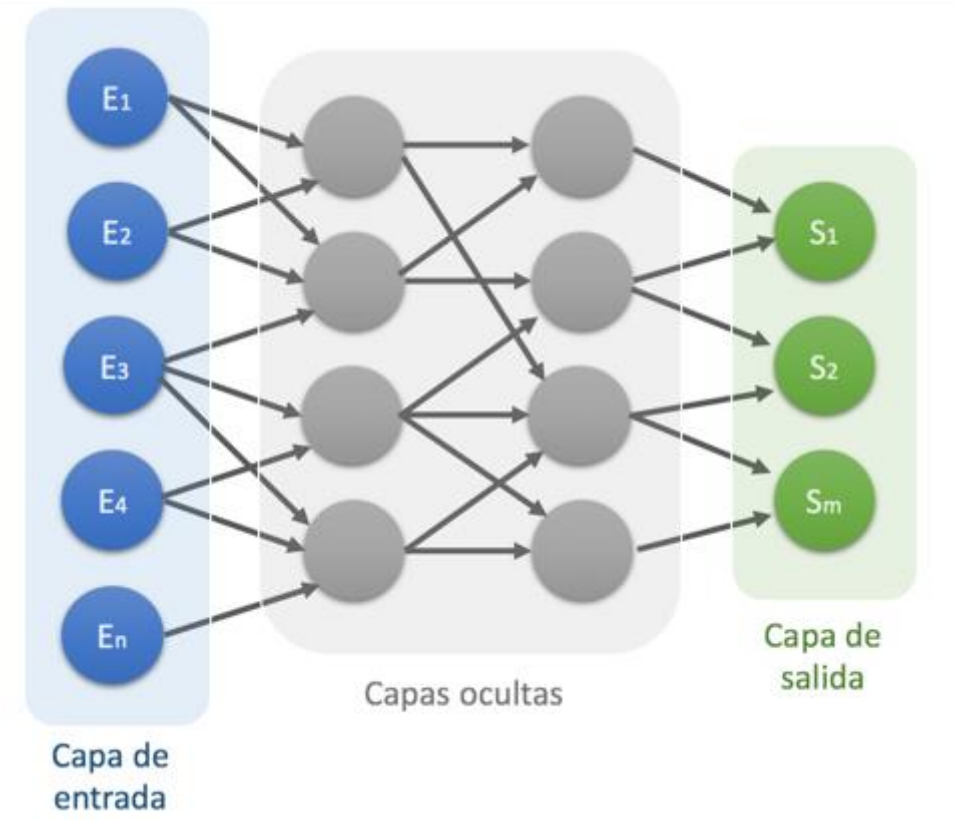

Fig. 2.8 Ilustración red neuronal [14]

En una red neuronal, el número de nodos de la capa de entrada debe corresponderse con el número de datos introducidos, mientras que el número de nodos dependerá del objetivo. Por ejemplo, en una regresión con una sola variable independiente, habrá solo un nodo de salida; mientras que en un problema de clasificación múltiple la capa de salida tendrá tantos nodos como categorías. Las capas ocultas pueden tener un tamaño variable, pero las funciones de activación de sus nodos deben estar

programadas de forma que reciban el número de datos de entrada equivalente al número de nodos de la capa anterior.

De igual modo que otros algoritmos, las funciones de activación utilizan datos de entrada (x), datos de salida (y), predicciones () y los parámetros β, que en las redes neuronales se encuentran en las conexiones entre nodos, como se ilustra en la siguiente imagen:

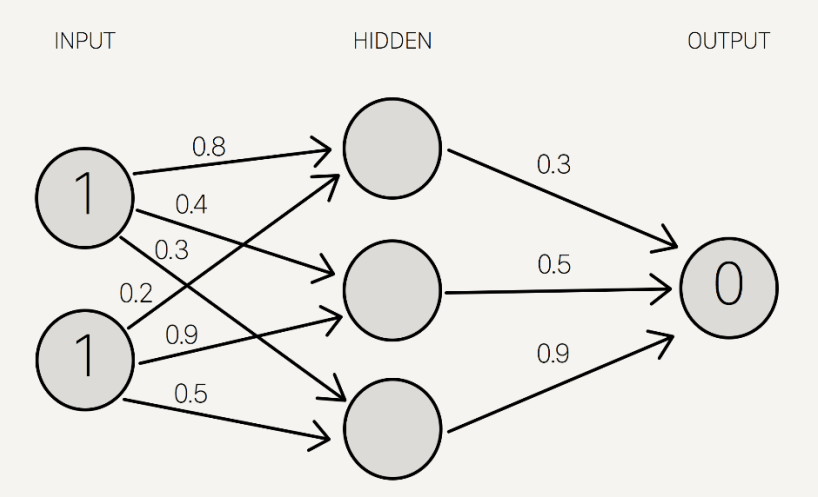

Fig. 2.9 Red neuronal con 1 capa [15]

Por tanto, para mejorar las predicciones de la red neuronal, estos parámetros (llamados *weights* en inglés) deben actualizarse para reducir la función de error, al igual que en los algoritmos previamente estudiados. Este proceso tiene dos pasos (que se repiten en cada iteración), *feedforward* (alimentación) y *backpropagation* (retroalimentación). Feedforward consiste en hacer pasar los datos por todas las capas hasta obtener el output (la predicción), una vez hechas las predicciones, se introducen en la función de error. Backpropagation consiste en propagar ese error calculado hacia las capas previas, actualizando los parámetros para reducir el error. En casos anteriores, para cambiar los valores de β bastaba con utilizar gradient descent, que empleaba el gradiente de la función de error, pero al haber varias capas, es necesario utilizar la regla de la cadena.

Las redes neuronales son un algoritmo particularmente potente ya que tienen la capacidad de encontrar relaciones entre datos con mínima supervisión humana. En muchos problemas de aprendizaje automático las variables básicas (x_1, x_2, x_3…) no son suficientes para crear un modelo con un error bajo. En estos casos, es necesario generar nuevas variables a través de la combinación de variables básicas: $x_1 \cdot x_2$, x^2_1, $x_2 . x_3 . x_4$, etc. Encontrar combinaciones de variables con capacidad predictiva de forma manual es difícil, pero las redes neuronales con múltiples capas (en un proceso conocido como *deep learning* o aprendizaje profundo) realizan este tipo de combinaciones de forma eficiente, permitiendo encontrar relaciones entre variables que otros algoritmos no pueden.

2.5. Convolutional Neural Networks (CNNs)

Las CNNs (redes neuronales convolucionales) son una variante de las redes neuronales aplicadas principalmente al reconocimiento y clasificación de imágenes. Antes de que se desarrollaran las CNNs, el método para clasificar imágenes consistía

en transformar la matriz de píxeles de una imagen en un vector, un proceso denominado *flattening* (aplanamiento), este vector se utilizaba como dato de entrada en las redes neuronales normales. De esta forma era posible clasificar imágenes simples o procesadas para tener la misma configuración espacial. Sin embargo, para imágenes no procesadas o casos más complicados este proceso no era efectivo, ya que era incapaz de identificar relaciones espaciales.

Las CNNs resuelven este problema mediante el uso de filtros (también llamados kernels) que transforman las imágenes de forma que se puedan procesar sin perder información importante. Los filtros son matrices que realizan la operación de convolución, multiplicando los valores de la imagen original para obtener las principales características de la imagen. Según los parámetros de la matriz filtro, se obtendrán un tipo de características u otras (bordes de las figuras, color, etc.), por lo que se suelen utilizar varios filtros para evitar perder información relevante. Es importante tener en cuenta el tamaño de la entrada y la salida de estas capas, ya que a diferencia de las redes neuronales normales (que solo tienen 2 dimensiones), en estas hay 4: número de imágenes, altura, anchura y profundidad. Las capas que utilizan filtros se denominan capas convolucionales, y a su salida, *convolved feature* (característica convolucionada).

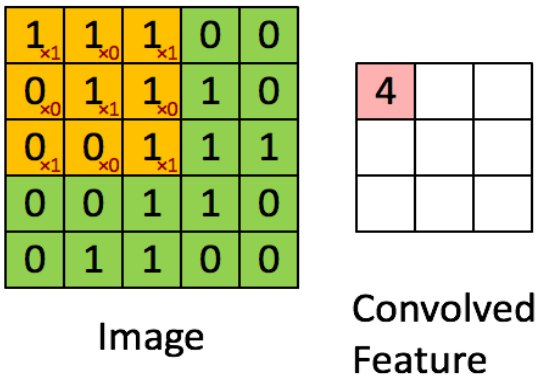

Fig. 2.10 CNN Kernel [16]

Además de las convolucionales (y de la entrada y salida), hay otros 3 tipos de capa en las CNNs: *Pooling layers* (capas de acumulación)*, Fully-connected layers* (capas completamente conectadas) y una capa de flattening. Las pooling layers tienen como objeto reducir las dimensiones de las convolved features para acelerar su computación, y pueden funcionar de dos formas: tomando la media de los valores de cada zona (*Average pooling)* o tomando el valor máximo (*Max pooling)*. Además, el Max pooling permite reducir el ruido (al descartar valores menos importantes), por lo que es la opción más usada.

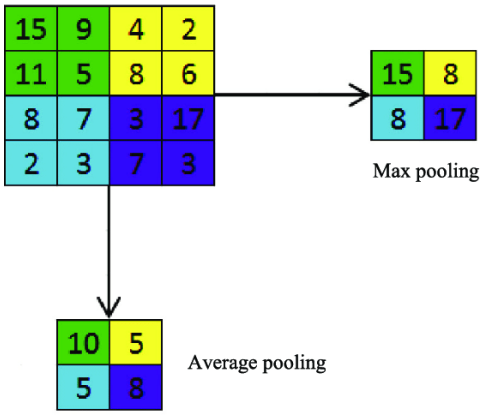

Fig. 2.11 Max vs Average pooling [17]

La capa de flattening se utiliza para poder traducir la salida de las capas convolucionales y de pooling a las capas fully-connected o de salida. Finalmente, las capas fully-connected tienen la estructura de las redes neuronales normales, y utilizan funciones como ReLU, permitiendo encontrar relaciones que las otras partes de la CNN no pueden.

3. CASOS DE ESTUDIO Y RESULTADOS

3.1. Introducción

En esta parte del proyecto se han escogido tres bases de datos como ejemplos para aplicar los métodos descritos en el Capítulo 2. La autoría de los datos originales, así como un enlace directo para acceder a los datos en internet han sido citados en la bibliografía.

Todos los experimentos han sido programados en Python, y el código utilizado se ha subido a GitHub para permitir su reproducción. [38]

Los casos prácticos estudiados y los métodos utilizados para resolverlos se resumen en la siguiente tabla (3.1):

TABLA 3.1
RESUMEN CASOS REALES ESTUDIADOS

BASE DE DATOS	TIPO DE PROBLEMA	ALGORITMOS EMPLEADOS
Concrete Compressive Data Set [18]	Regresión	Regresión Lineal Kernel-Ridge Regression Red Neuronal
HTRU2 Data Set [20]	Clasificación	Regresión Logística Support Vector Machines Red Neuronal
Fashion-MNIST Data Set [22]	Clasificación de imágenes	Red Neuronal Convolucional (CNN)
Breast Cancer Wisconsin (Diagnosis) Data Set [41]	Reducción Dimensional	PCA

3.2. Concrete Compressive Strength Data Set

Esta base de datos [18] contiene información sobre el hormigón y nos permite analizar la relación entre los componentes usados en la mezcla y el esfuerzo de comprensión (en MPa), que es la característica más importante para diseñar un hormigón de calidad y es la variable de salida u objetivo. Hay 8 variables independientes, que son la edad (en días) y los componentes (en Kg/m^3): cemento, escoria de alto horno, ceniza volátil, agua, aditivo superplastificante, áridos finos y áridos gruesos. Esta base de datos contiene 1030 observaciones.

Se tratará este ejemplo como un caso de regresión, y se emplean 3 algoritmos para resolverlo: regresión linear, kernel-ridge regression y una red neuronal. La red neuronal tiene 5 capas cuyas funciones de activación son lineares (en 4 capas) y tangente hiperbólica (en 1), mientras que su función de error es el error cuadrático medio (ECM). Para entrenar la red neuronal se realizaron 200 iteraciones, punto en el que ocurre la convergencia.

Para evitar el overfitting se utilizan dos métodos de preprocesamiento: la normalización de los datos y la división de la base de datos en dos subconjuntos. La

normalización consiste en transformar los datos de las variables para que tengan una escala común, esto es necesario ya que la edad tiene un rango de valores completamente distinto a las demás variables. La normalización permite que el algoritmo trate todas las variables de la misma forma y mejora el proceso de entrenamiento.

Por otro lado, se divide la base de datos en dos subconjuntos, *training set* (set de entrenamiento) y *test set* (set de prueba), lo que permite comprobar que el modelo generaliza bien y no se ha ajustado de forma excesiva a los datos de entrenamiento. Si la función de error tiene un valor similar para ambos sets, significa que no ha ocurrido overfitting.

Para comparar la calidad de los modelos se emplean tres medidas diferentes: *Mean* Absolute Error (error absoluto medio), ECM y R^2. ECM, al ser la función de error (*loss function)* ha sido calculada para ambos subconjuntos: test y training. La medida R^2 también se conoce como coeficiente de determinación, y refleja la proporción de varianza del output (variable dependiente) que el modelo es capaz de predecir a partir de las variables independientes. Existen varias definiciones matemáticas distintas, en este caso se ha utilizado la implementación incluida en la librería scikit-learn. En esta implementación los valores de R^2 tienen un valor máximo de 1.0 (que significa que el modelo es perfecto), mientras que un valor de 0.0 equivale a un modelo que prediga una salida constante (ignorando las variables de entrada), un valor negativo indica que el modelo es peor que predecir la constante. [19]

TABLA 3.2
COMPARACIÓN MODELOS PARA BASE DE DATOS HORMIGÓN

MODELO	R2 SCORE	MEAN ABSOLUTE ERROR	TRAINING SET LOSS (ECM)	TEST SET LOSS (ECM)
Regresión Linear	0.648	7.73	109.7	99.3
Kernel Ridge Regression	0.649	7.79	110.2	98.9
Red Neuronal	0.002	13.05	272.1	280.2

En este caso, los modelos de regresión lineal y kernel ridge regression han obtenido resultados muy similares en todas las medidas, y con un valor alto de R^2 que indica que tienen buena capacidad predictiva. Por otro lado, el modelo de red neuronal ha tenido un error mayor y un valor muy bajo de R^2, por lo que apenas tiene poder predictivo, este modelo debe ser descartado o modificado para obtener mejores resultados. Además, se puede observar que en todos los modelos el error obtenido en ambos subconjuntos (test y training) es muy parecido, por lo que se puede afirmar que no ocurre overfitting.

3.3. HTRU2 Data Set

Esta base de datos [20] contiene 17898 observaciones de estrellas, siendo el objetivo determinar cuáles de estas observaciones son púlsares (un tipo de estrella de

neutrones con radiación y un poderoso campo magnético). Las observaciones contienen 9 atributos, de las cuales 8 son variables cuantitativas continuas (que son las variables independientes); mientras que la última indica la clase y toma únicamente los valores 1 (en caso de ser un púlsar) y 0 (si no lo es). La clase es la variable independiente, por lo que este es un problema de clasificación binaria. Se puede observar que la mayoría de las observaciones en la base de datos no son púlsares:

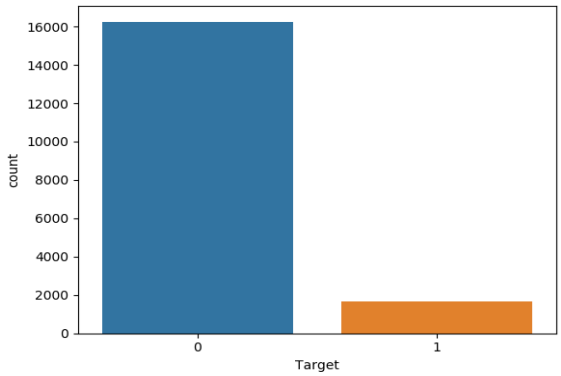

Fig. 3.1 Número de observaciones por clase

Los algoritmos utilizados para resolver el problema son: regresión logística, support vector machines y una red neuronal. La red neuronal utilizada tiene cuatro capas, de las cuales tres tienen una función de activación linear, mientras que la capa de salida tiene una función sigmoide, lo cual es necesario al ser un problema de clasificación. La función de error utilizada es *Cross-Entropy Loss* (Entropía cruzada) que tiene en cuenta los valores de probabilidad y no sólo la predicción binaria, por lo que el resultado de la función no depende de la decisión boundary escogida.

Al igual que en el anterior ejemplo, se separa la base de datos en training y test set para evitar el overfitting. Además, en la red neuronal se utiliza la técnica de *dropout* (que desactiva algunos nodos al azar), que también ayuda a evitar el overfitting.

Para comparar la calidad de los modelos se utilizan varios métodos, el primero de ellos es el conocido como matriz de confusión. A partir de los valores reales de las observaciones, las predicciones del modelo (tomando una decision boundary de 0.5, es decir, 50% de probabilidad) se clasifican de la siguiente forma:

		Predicción	
		Positivos	**Negativos**
Observación	**Positivos**	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos (FP)	Verdaderos Negativos (VN)

Fig. 3.2 Matriz de Confusión [21]

A partir de la matriz de confusión se desarrollan varias medidas que permiten comparar el funcionamiento del modelo según distintas prioridades. Algunas de estas son:

- **Tasa de error**: indica el porcentaje de veces que se equivoca el modelo. Se calcula dividiendo la suma de falsos positivos más falsos negativos entre el total de observaciones.
- **Tasa de verdaderos positivos:** también conocido como exhaustividad (o *recall* en inglés*),* es el porcentaje de casos positivos que ha identificado el modelo. Su fórmula es:

$$Recall = \frac{VP}{VP+FN} \tag{3.1}$$

- **Tasa de falsos positivos:** equivale al número de falsos positivos entre el total de casos negativos.
- **Tasa de verdaderos negativos:** equivale al número de verdaderos negativos entre el total de casos negativos.
- **Precisión:** indica la tasa de acierto entre los casos que el modelo ha predicho como positivos. Su fórmula es:

$$Precisión = \frac{VP}{VP+FP} \tag{3.2}$$

Aunque estas medidas aportan información detallado sobre los aciertos y fallos del modelo, no proporcionan una imagen general de qué modelo es superior. Por este motivo se utiliza la F_1 *Score* (también conocida como Valor-F) que combina precisión y recall y se suele utilizar para comparar modelos. Se calcula de la siguiente forma:

$$F_1 = 2\frac{Precisión \times Exhaustividad}{Precisión + Exhaustividad} \tag{3.3}$$

A continuación, se comparan las matrices de confusión de cada uno de los modelos utilizados:

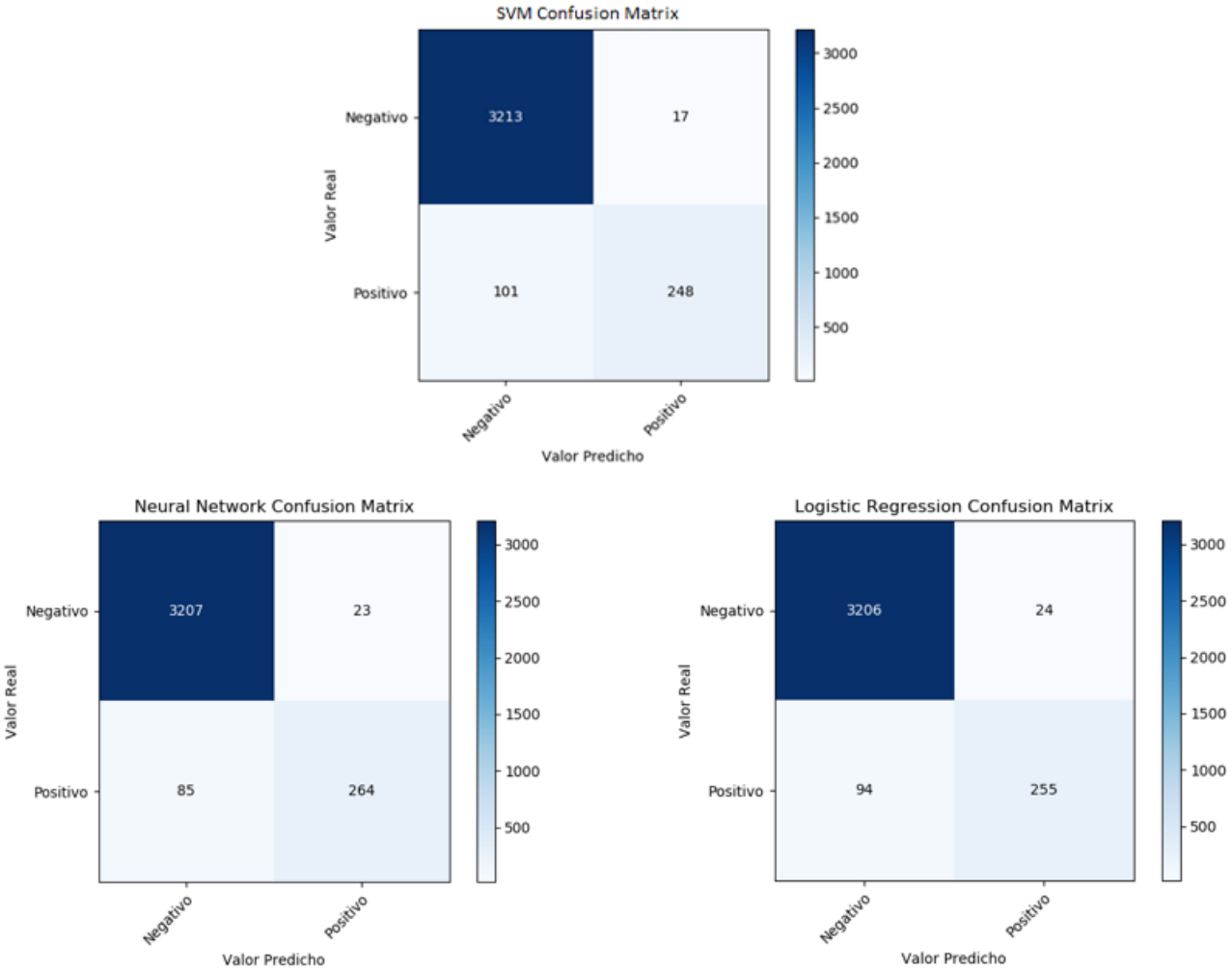

Fig. 3.3 Matrices de Confusión para modelos de la base de datos HTRU2

A simple vista, las tres matrices son similares, y se puede observar, por ejemplo, que el modelo SVM tiene a predecir valores negativos más que los demás modelos, ya que tiene cantidades mayores de tanto verdaderos negativos como falsos negativos. Sin embargo, es complicado sacar conclusiones sobre cuál de los tres modelos es superior, por lo que es necesario calcular las medidas mencionadas previamente para identificar el mejor modelo. Se encuentran en la siguiente tabla (3.3):

TABLA 3.3
COMPARACIÓN MODELOS BASE DE DATOS HTRU2

MODELO	F1 SCORE	TASA DE ERROR	TASA VERDADEROS POSITIVOS	TASA FALSOS POSITIVOS	TASA VERDADEROS NEGATIVOS	RECALL
SVM	0.799	0.033	0.711	0.005	0.995	0.914
Red Neuronal	0.836	0.030	0.756	0.007	0.993	0.936
Regresión Logística	0.815	0.033	0.731	0.007	0.993	0.919

Las medidas calculadas son de nuevo similares, lo que permite observar la utilidad del F_1-Score, que sí es capaz de diferenciar mejor los modelos. En concreto, la red neuronal es el algoritmo que mejor consigue resolver el problema, y destacando en particular en su tasa de verdaderos positivos. La regresión logística y SVM funcionan algo peor, y se diferencian en que la regresión logística tiene mayor exhaustividad (siendo capaz de encontrar más verdaderos positivos) pero a la vez comete más errores. El modelo SVM predice casos positivos de forma menos agresiva que los otros modelos, como se aprecia también en su matriz de confusión, lo que sería útil en casos en los que fuera más importante evitar falsos positivos.

Hay que tener en cuenta que todas las medidas mostradas previamente han sido calculadas para una única decision boundary (0.5), pero es posible que exista un mejor valor límite para las predicciones. Como calcular y comparar las medidas de forma manual es difícil, existe un método que permiten comparar de forma visual, la denominada *Precision-Recall Curve* (Curva de Precisión-Exhaustividad). Esta curva permite tomar decisiones basadas según la principal prioridad del usuario: en caso de que sea más importante encontrar todos los positivos se escoge mayor recall, mientras que si es más importante evitar los falsos positivos la prioridad será la precisión. En caso de que se busque una medida única de cuál modelo es mejor, se puede calcular el área bajo la curva, ya que un mayor valor indica que la precisión se ha mantenido alta al aumentar la exhaustividad. Sin embargo, se suele utilizar una alternativa más sencilla de calcular, la precisión media (*Average Precision* o AP), que se calcula como la media de los valores de precisión (evaluados para cada punto en el que aumenta la tasa de recall).

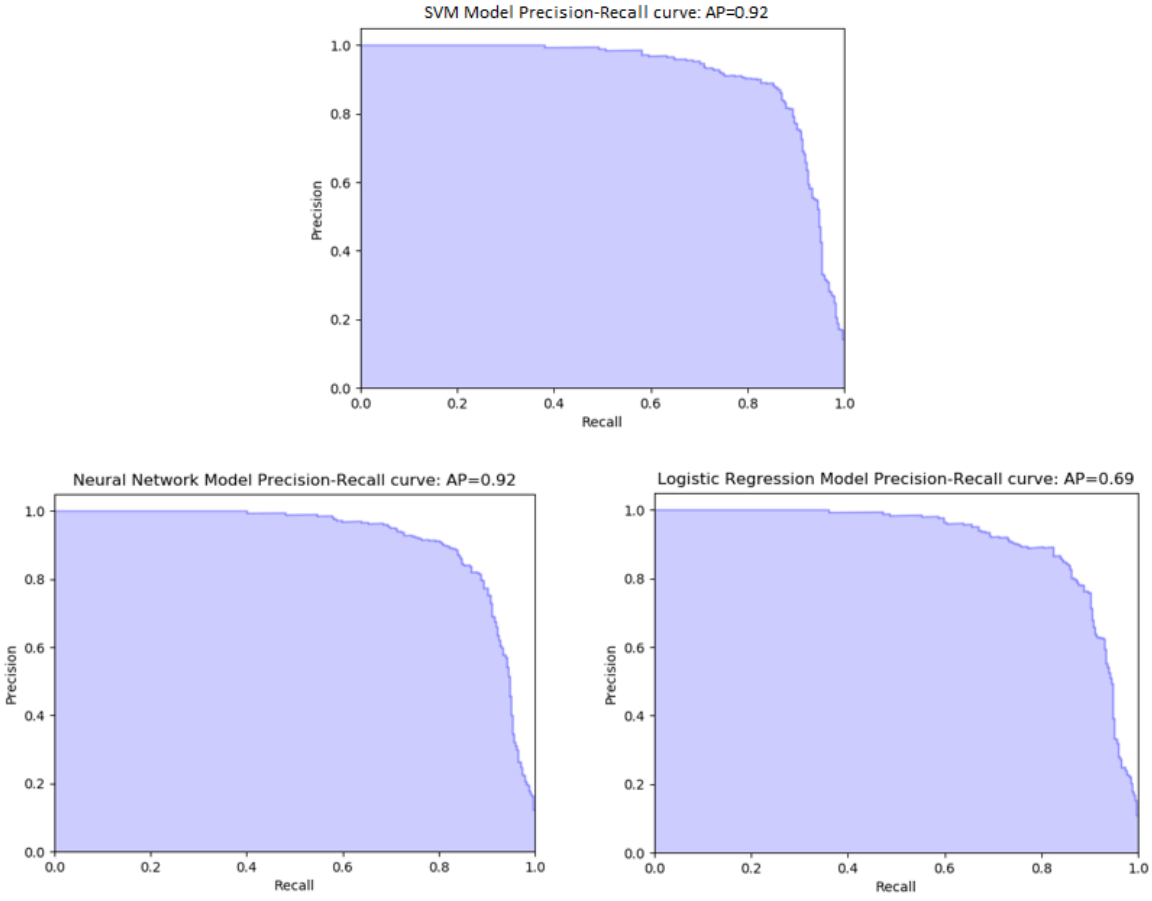

Fig. 3.4 Precision-Recall Curves para base de datos HTRU2

Comparando las figuras y la precisión media se puede observar que, aunque a simple vista los modelos son similares, la regresión logística pierde precisión de forma rápida cuando aumenta el recall, teniendo un valor de AP más bajo que los otros modelos, por lo que es menos adecuado.

3.4. Fashion-MNIST Data Set

La base de datos MNIST original contiene miles de imágenes de dígitos escritos a manos, y es ampliamente usada en el campo del aprendizaje automático como base para probar modelos y nuevos algoritmos. Muchos investigadores han intentado desarrollar modelos perfectos para esta base de datos, y aunque todavía no se ha conseguido, varios modelos han conseguido una tasa de error por debajo de 1%. Los modelos que mejores resultados han obtenido han sido las redes neuronales convolucionales (CNNs).

Sin embargo, MNIST es usada de forma excesiva, además de ser demasiado fácil, según investigadores como François Chollet. [23] Por estos motivos, en este proyecto se utiliza una variante llamada Fashion-MNIST, que contiene 70000 imágenes de artículos de ropa (60000 en el training set y 10000 en el test set). Las imágenes tienen un tamaño de 28x28 píxeles y son todas en blanco y negro. [22] Ejemplos de las imágenes que se pueden encontrar en esta base de datos se pueden ver en la siguiente figura 3.5:

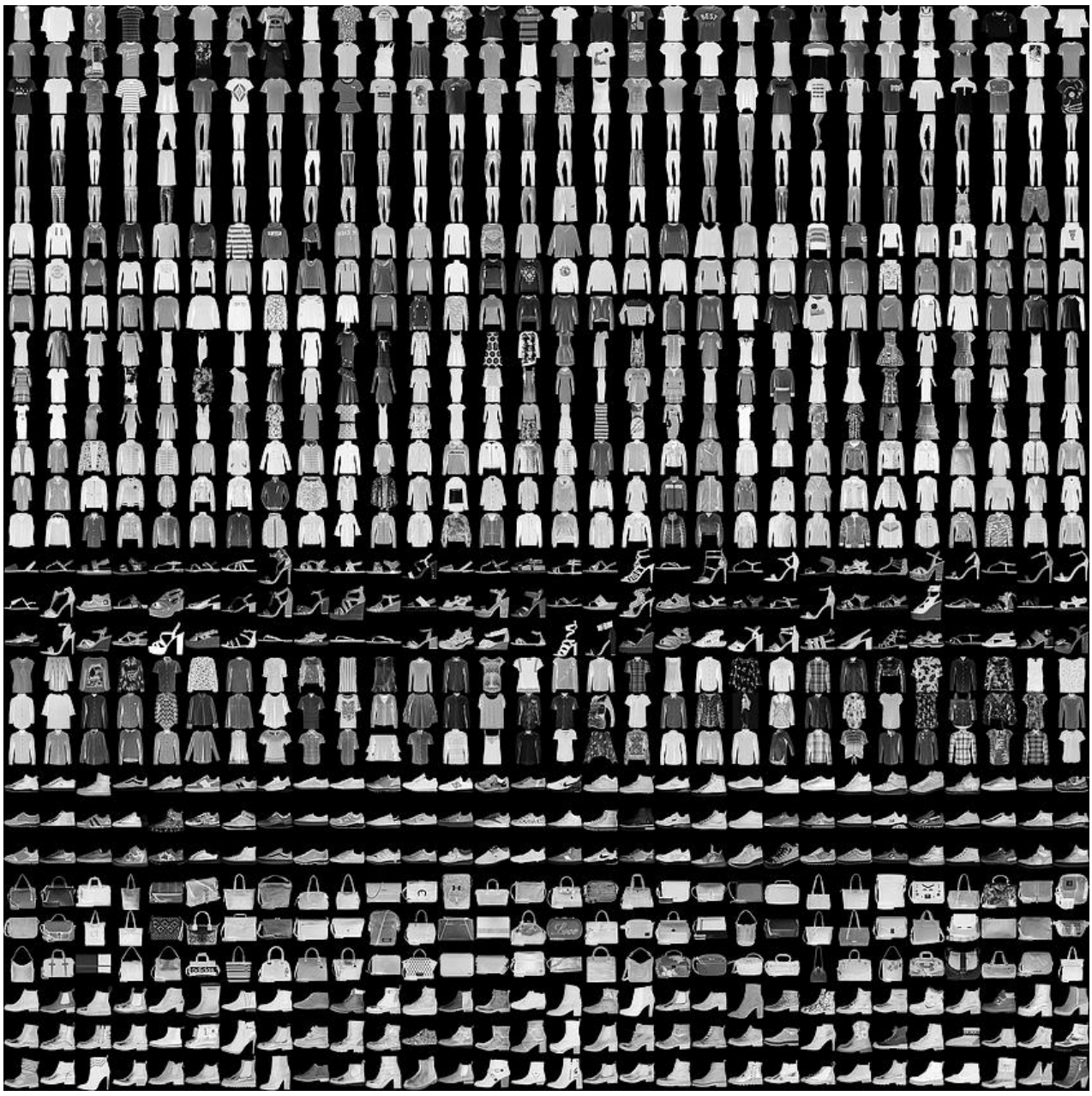

Fig. 3.5 Ejemplo de imágenes Fashion-MNIST [22]

Las imágenes de Fashion-MNIST se dividen a su vez en 10 clases diferentes: *T-shirt* (camiseta)*, Trouser* (pantalón)*, Pullover* (suéter)*, Dress* (vestido)*, Coat* (abrigo)*, Sandal* (sandalia)*, Shirt* (camisa)*, Sneaker* (zapatilla)*, Bag* (bolso)*, Ankle boot* (bota). Por tanto, se trata de un problema de clasificación múltiple de reconocimiento de imágenes, para los cuales las redes neuronales convolucionales son el método con mayor potencial.

La red neuronal diseñada tiene 9 capas: 3 convolucionales (con función ReLU añadida para acelerar la computación), 2 capas de max pooling, 1 capa de flattening y finalmente 3 capas fully-connected (con función linear) entre las que se incluye la capa de salida. Además, se ha utilizado el método de dropout para la primera capa fully-connected.

Para entrenar el modelo se ha separado el training set en dos subconjuntos: training set y *validation set* (set de validación), como forma de evitar el overfitting y seleccionar el mejor modelo posible y, además, el training set se divide en lotes de 20 imágenes. En cada iteración se introducen estos lotes uno por uno, y el error (del training set) de cada lote se utiliza para actualizar el modelo. Una vez han pasado todos los lotes por el modelo, acaba la iteración y se calcula el error del set de validación. Este error del set de validación se guarda para todas las iteraciones, y se escoge como modelo final el que tenga el valor mínimo de error de validación. Finalmente, se prueba el modelo en el test set.

Los resultados obtenidos tras 15 iteraciones (para las que se ha tardado 30 minutos en calcular) son:

TABLA 3.4
RESULTADOS DEL MODELO PARA FASHION-MNIST

CLASE	N.º DE ACIERTOS / N.º TOTAL	PRECISIÓN
T-shirt/top	819/1000	81%
Trouser	979/1000	97%
Pullover	872/1000	87%
Dress	871/1000	87%
Coat	832/1000	83%
Sandal	970/1000	97%
Shirt	698/1000	69%
Sneaker	965/1000	96%
Bag	972/1000	97%
Ankle boot	962/1000	96%
TOTAL	**8940/1000**	**89%**

El modelo en general tiene una precisión alta, superior al 80% para todas las categorías excepto *Shirt* (camisa). Probablemente esto se debe al parecido entre las camisas y camisetas, por lo que para mejorar el modelo sería necesario descubrir si el problema está en las capas convolucionales o en las lineales. Como es difícil entender lo que ocurre dentro de una red neuronal convolucional con tantas capas, será necesario tantear hasta encontrar un modelo mejor. También es posible encontrar pistas sobre el problema comparando algunas de las imágenes de la base de datos, para entender si hay más diferencia entre detalles o si el problema es que el modelo no entiende las relaciones espaciales entre partes de la imagen.

3.5. Breast Cancer Wisconsin (Diagnosis) Data Set

Esta base de datos [41] contiene 569 ejemplos de tumores, clasificados según si son malignos (letra 'M') o benignos ('B'), que sería la variable objetivo en un problema de clasificación. Además, hay 30 variables dependientes numéricas, que describen los núcleos de células presentes en las imágenes utilizadas para estudiar los tumores. Entre las observaciones de la base de datos existen 357 casos benignos y 212 malignos.

Para resolver este problema se podrían utilizar técnicas de clasificación como en el caso anterior, pero al haber un gran número de variables y baja cantidad de observaciones de la base de datos, los métodos de clasificación podrían tener problemas para separar los casos. Por tanto, se intentará realizar una reducción de dimensiones primero. Dado que es un problema de clasificación binario y que hay pocos ejemplos, se aplicará el método PCA con el objetivo de reducir las variables dependientes a 2 vectores (componentes principales), lo que además permite visualizar los casos en una gráfica única.

Como PCA es un método de aprendizaje no supervisado, antes de introducir los datos en el algoritmo hay que procesarlos para quitar la variable objetivo que indica si son malignos o benignos. Además, para el algoritmo trate todas las variables de la misma forma se han normalizado todas para que estén a la misma escala. El resultado obtenido tras aplicar PCA se puede observar en la figura 3.6:

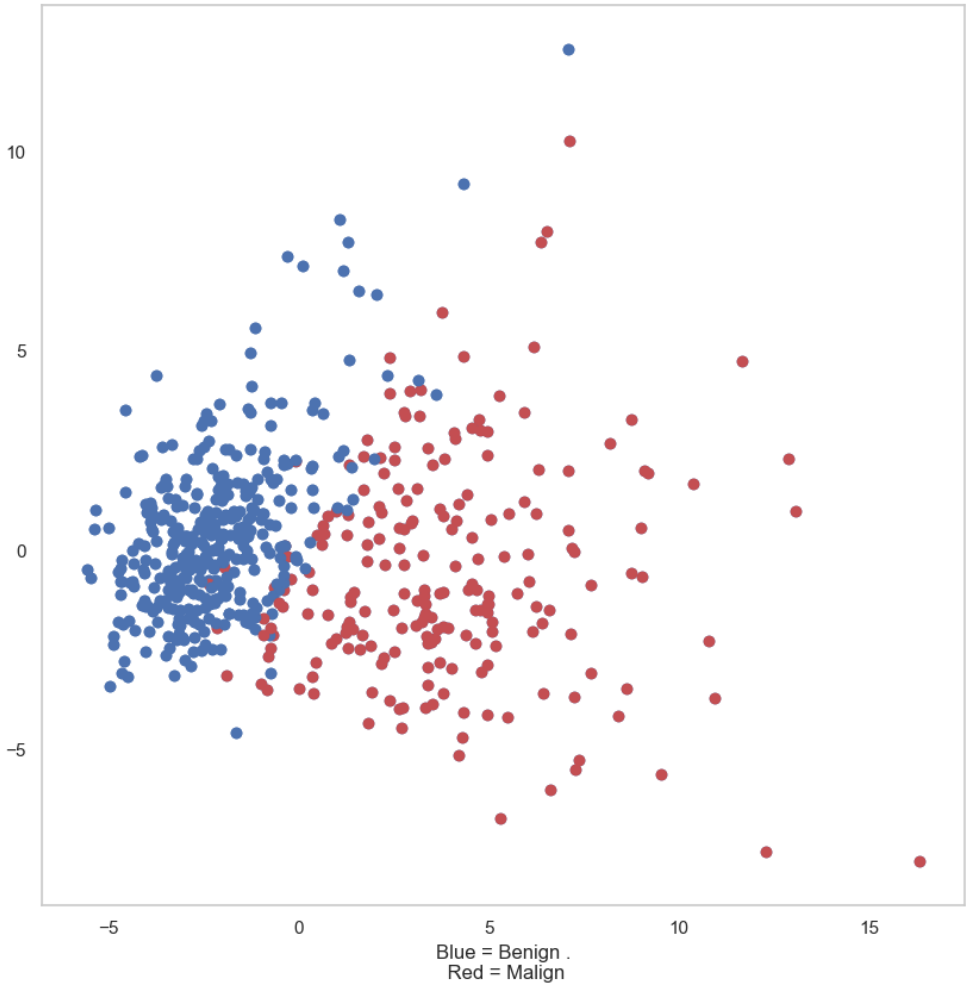

Fig. 3.6 Resultado método PCA en caso práctico Wisconsin Breast Cancer

El resultado obtenido es satisfactorio, ya que las nuevas variables obtenidas permiten separar los casos malignos de los benignos. Además de permitir su visualización, los nuevos datos obtenidos podrían ser procesados con un algoritmo de clasificación para hacer predicciones de nuevos casos.

4. CONCLUSIONES

4.1. Discusión sobre los resultados

En el campo de la ciencia de datos existe una gran cantidad de algoritmos, por lo que, al intentar resolver un problema, puede ser difícil escoger el algoritmo adecuado. Esto provoca que muchos investigadores y sobre todo estudiantes escojan simplemente los modelos más populares, como las redes neuronales, o practiquen con bases de datos de prueba excesivamente utilizadas como MNIST. Sin embargo, esta toma de decisiones puede causar problemas, como se ilustra en los ejemplos de este proyecto.

En la primera base de datos utilizada (Concrete Compressive Strength Dataset), el algoritmo que peor funciona es la red neuronal, siendo sus predicciones poco mejores que predecir siempre la misma constante. Sin embargo, los otros modelos funcionan mucho mejor y consiguen resolver el problema de forma satisfactoria. Posiblemente esto se debe a que esta base de datos contiene pocas observaciones (solo 1030, en comparación con los 17898 y 70000 de las otras bases de datos estudiadas), lo que muestra uno de los problemas de las redes neuronales, la necesidad de una gran cantidad de datos para funcionar correctamente. La tercera base de datos muestra también este problema de forma diferente, ya que fue necesario mucho tiempo para entrenar el modelo, lo que refleja que no sólo es importante diseñar un buen modelo, sino que se debe tener en cuenta la capacidad del hardware utilizado.

Además, existen otros problemas asociados al uso de algoritmos complejos como su falta de transparencia, que no solo hace difícil encontrar el problema en caso de que no funcionen bien, sino que también provoca dificultades cuando se desea descubrir qué variables influyen más en el resultado. En el caso del hormigón, aunque no hay mucha diferencia entre las predicciones del modelo de kernel-ridge regression y el de regresión linear, el segundo es más útil ya que permite encontrar las variables que más influyen en el resultado final de forma más sencilla. Como ejemplo, algunas compañías como ANZ mencionan este problema:

“In a deep-learning environment, it becomes very difficult to work out the factors that were the most predictive for this instance, or for this customer. Before we roll out any deep-learning models, we need to solve for that -- even though it's not legislated here. I think it is good practice to be able to know why decisions are being made.” [25]

Por estos motivos, es necesario que los científicos de datos sean capaces de utilizar una gran variedad de algoritmos en lugar de aprender únicamente los más complejos o populares.

4.2. Líneas futuras de trabajo

Aunque se han mencionado en la parte teórica y aplicado en la red neuronal convolucional y en el problema de PCA, no se han realizado problemas para todos los métodos comunes de aprendizaje no supervisado. Por tanto, entre los próximos objetivos futuros serán buscar una base de datos ideal para poner a prueba algoritmos de clustering y de detección de anomalías.

Este trabajo sirve de introducción a la ciencia de datos y específicamente al aprendizaje automático, pero es importante destacar que este no es el único campo existente dentro de la ciencia de datos. Este trabajo principalmente es una introducción, pero para la mayoría de los trabajos es necesaria una especialización. Por ejemplo, a la hora de utilizarlos en empresas o presentarlos al público la capacidad de presentar los resultados de forma atractiva y clara es crucial. En esto se enfoca la disciplina de visualización de datos, y existen varias herramientas muy utilizadas y demandadas por las empresas, como Tableau o el lenguaje de programación R.

Otro conocimiento necesario es el manejo de bases de datos y de las herramientas comunes empleadas para este fin como SQL. Estos conocimientos son muy importantes en el ámbito empresarial, ya que muchas compañías manejan grandes cantidades de datos y modelos que deben ser mantenidos correctamente. Además, existen múltiples tecnologías que un científico debe conocer, como Spark, Hadoop o Mongo DB, las cuales tienen múltiples aplicaciones en el manejo y procesamiento de los datos y cuyo uso está muy extendido en la industria.

BIBLIOGRAFÍA

[1] Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.

[2] https://www.forbes.com/sites/tomtaulli/2019/05/04/how-to-reskill-your-workforce-for-ai-artificial-intelligence/#36891c513eb9 *Consultado el 12 de Junio de 2019.*

[3] https://www.boe.es/buscar/pdf/2018/BOE-A-2018-16673-consolidado.pdf *Consultado el 5 de Junio de 2019.*

[4] https://eur-lex.europa.eu/legal-content/EN/TXT/?qid=1528874672298&uri=CELEX%3A32016R0679 *Consultado el 5 de Junio de 2019.*

[5] https://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html *Consultado el 6 de Junio de 2019.*

[6] https://scikit-learn.org/stable/modules/kernel_ridge.html *Consultado el 10 de Junio de 2019.*

[7] https://scikit-learn.org/stable/modules/linear_model.html#ridge-regression *Consultado el 10 de Junio de 2019.*

[8] https://www.researchgate.net/figure/Decision-Boundary-at-P-5-of-the-logistic-regression-model-The-line-represents-the_fig3_230742568 *Consultado el 7 de Junio de 2019.*

[9] https://christophm.github.io/interpretable-ml-book/logistic.html *Consultado el 7 de Junio de 2019.*

[10] John D. Hunter. Matplotlib: A 2D Graphics Environment, Computing in Science & Engineering, 9, 90-95 (2007),DOI:10.1109/MCSE.2007.55

[11] https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 *Consultado el 8 de Junio de 2019.*

[12] https://home.deib.polimi.it/matteucc/Clustering/tutorial_html/ *Consultado el 8 de Junio de 2019.*

[13] https://home.deib.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html *Consultado el 8 de Junio de 2019.*

[14] http://www.diegocalvo.es/definicion-de-red-neuronal/ *Consultado el 8 de Junio de 2019.*

[15] https://stevenmiller888.github.io/mind-how-to-build-a-neural-network/ *Consultado el 8 de Junio de 2019.*

[16] https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 *Consultado el 9 de Junio de 2019.*

[17] https://www.researchgate.net/figure/Average-versus-max-pooling_fig1_317496930 *Consultado el 9 de Junio de 2019.*

[18] I-Cheng Yeh, "Modeling of strength of high performance concrete using artificial neural networks," Cement and Concrete Research, Vol. 28, No. 12, pp. 1797-1808 (1998). https://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength *Consultado el 9 de Junio de 2019.*

[19] https://scikit-learn.org/stable/modules/model_evaluation.html#r2-score *Consultado el 9 de Junio de 2019.*

[20] R. J. Lyon, B. W. Stappers, S. Cooper, J. M. Brooke, J. D. Knowles, Fifty Years of Pulsar Candidate Selection: From simple filters to a new principled real-time classification approach, Monthly Notices of the Royal Astronomical Society 459 (1), 1104-1123, DOI: 10.1093/mnras/stw656 https://archive.ics.uci.edu/ml/datasets/HTRU2 *Consultado el 9 de Junio de 2019.*

[21] https://rpubs.com/chzelada/275494 *Consultado el 10 de Junio de 2019.*

[22] https://research.zalando.com/welcome/mission/research-projects/fashion-mnist/ *Consultado el 10 de Junio de 2019.*

[23] https://twitter.com/fchollet/status/852592598128615424 *Consultado el 10 de Junio de 2019.*

[24] Han Xiao and Kashif Rasul and Roland Vollgraf, Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms, arXiv, cs.LG/1708.07747

[25] https://www.zdnet.com/article/anz-bank-unpicking-neural-networks-in-effort-to-avoid-dangers-of-deep-learning/ *Consultado el 10 de Junio de 2019.*

[26] Wes McKinney. Data Structures for Statistical Computing in Python, Proceedings of the 9th Python in Science Conference, 51-56 (2010)

[27] Christopher M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics), Springer-Verlag New York, Inc., Secaucus, NJ, 2006

[28] Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, The MIT Press, 2016

[29] Kevin P. Murphy, Machine Learning: A Probabilistic Perspective, The MIT Press, 2012

[30] https://medium.com/activewizards-machine-learning-company/top-8-data-science-use-cases-in-manufacturing-749256b8f1ee *Consultado el 12 de Junio de 2019.*

[31] https://www.smart-energy.com/industry-sectors/smart_water/data-science-water-industry-black-veatch/ *Consultado el 12 de Junio de 2019.*

[32] https://www.infosecurity-magazine.com/blogs/data-science-helping-cybersecurity-1/ *Consultado el 12 de Junio de 2019.*

[33] https://www.lavanguardia.com/vida/20190604/462681041614/orange-aboga-por-la-transformacion-digital-de-las-fabricas-del-futuro-como-factor-de-crecimiento-economico.html *Consultado el 12 de Junio de 2019.*

[34] https://www.globenewswire.com/news-release/2019/06/11/1867093/0/en/Altran-expands-European-Data-Science-and-AI-capabilities.html *Consultado el 12 de Junio de 2019.*

[35] https://www.forbes.com/sites/louiscolumbus/2017/05/13/ibm-predicts-demand-for-data-scientists-will-soar-28-by-2020/#187e84ac7e3b *Consultado el 12 de Junio de 2019.*

[36] https://scikit-learn.org/stable/modules/metrics.html *Consultado el 13 de Junio de 2019.*

[37] http://www.statsoft.com/textbook/support-vector-machines *Consultado el 13 de Junio de 2019.*

[38] https://github.com/GermanGL/TFG-Ciencia-de-Datos *Consultado el 13 de Junio de 2019.*

[39] https://www.udacity.com/course/deep-learning-pytorch--ud188 *Consultado el 13 de Junio de 2019.*

[40] https://www.coursera.org/learn/machine-learning *Consultado el 13 de Junio de 2019.*

[41] W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on Electronic Imaging: Science and Technology, volume 1905, pages 861-870, San Jose, CA, 1993. https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic) *Consultado el 13 de Junio de 2019.*